Learning Hierarchical Models of Activity

Sarah Osentoski, Victoria Manfredi, Sridhar Mahadevan
Computer Science Department
University of Massachusetts
Amherst, MA 01003-9264
Email: {sosentos, vmanfred, mahadeva}@cs.umass.edu

***Abstract*— This paper investigates learning hierarchical statistical activity models in indoor environments. The Abstract Hidden Markov Model (AHMM) is used to represent behaviors in stochastic environments. We train the model using both labeled and unlabeled data and estimate the parameters using Expectation Maximization (EM). Results are shown on three datasets: data collected in lab, entryway, and home environments. The results show that hierarchical models outperform flat models.**

I. Introduction

Robots increasingly work and function within environments such as offices, museums, and hospitals that are also populated by human beings. In order to fully interact in this dynamic environment, it is necessary for them to understand the movement of other occupants throughout the environment. A first step toward this understanding of other agents, be they human or robotic, is the ability to recognize different activities based on movements throughout the space.

Much previous research in mobile robotics has explored localization and mapping given a static environment. Some of this work [1], [2] has studied methods to detect and track people within an environment. However, in this work the person is tracked based on a motion model of typical human movement: the intentions or the higher level tasks of the person are not modeled. Other work has been done that does model intention [3], [4]. This work clusters similar motion trajectories but does not model activity in a hierarchical manner.

Bui [5] introduces the Abstract Hidden Markov Model (AHMM) as a hierarchical statistical model for representing activities. In this work the AHMM was used as a probabilistic framework for plan recognition, where the model parameters were hand coded instead of learned. Murphy [6] investigated learning restricted 1-level AHMM models where the value of the top-level nodes were observed.

In this paper we use AHMMs [5] to model motion through indoor environments. We represent the model as a Dynamic Bayesian Network (DBN) and use Expectation Maximization (EM) to estimate the parameters of the DBN in order to learn behaviors. This work extends previous work by investigating what advantages different levels of hierarchy provide, and compares the performance of 1-level and 2-level AHMM models on both labeled and unlabeled training instances.

The rest of this paper is organized as follows. Section II reviews the basics of the AHMM. Section III describes algorithms learning the model. Section IV describes the datasets and methods. Section V presents the results and Section VI summarizes the contributions and describes future work.

II. Abstract Hidden Markov Model

The AHMM is a multi-scale statistical model for representing behaviors in stochastic, noisy situations. Hierarchy plays an important role in activities. Imagine a person exiting a building from a room on the second floor. This behavior can be broken into multiple sub-behaviors. A possible sub-behavior of the general behavior is to exit the second floor. A sub-behavior of this behavior could be to navigate to the stairway. As part of this behavior the person must leave the room. To do this, the person must take an action, in this case a step in the right direction. The AHMM provides a way to model this type of stochastic process and allows the robot to infer what the person is doing based on its observations at each timestep.

The AHMM provides a top down decomposition for a fixed behavior. A behavior maps a state to an action. The AHMM provides a method of modeling hierarchical behaviors. In hierarchical behaviors a high-level behavior can call a more refined low-level behavior according to some distribution. This low-level behavior will call a lower-level behavior. This process continues until the most primitive behavior possible is performed. In the domains with discrete actions the most primitive behavior would be a single action. When a low-level behavior terminates in some state then the parent behavior may also terminate with some probability so long as the current state is in the set of destination states of the parent behavior.

A. Representation

An AHMM can be represented as a DBN as in Figure 1. Edges between nodes represent dependencies. An AHMM has five different types of nodes, s_t, o_t, m_t, π_t^k, and e_t^k. Let s_t represent the state of the agent at time t. Since the true state of the agent is hidden, observations, o_t, which are a stochastic function of the state, are required. Observations are modeled as a mixture of Gaussians. m_t represents the mixture component of s_t. π_t^k represents the abstract behavior at level k and time t. e_t^k is the termination flag for π_t^k. $e_t^k =$ terminate signifies the natural completion of π_t^k.

B. Transition Model

Each node represents a conditional probability distribution (CPD) or table (CPT). The level 2 behavior at time t depends upon the level 2 behavior, state, and termination flag 2 at the previous timestep $t-1$. We define this as

$$P(\pi_t^2|\pi_{t-1}^2, s_{t-1}, e_{t-1}^2) =$$
$$\begin{cases} \sigma(s_{t-1}, \pi_t^2) & \text{if } e_{t-1}^2 = \text{terminate} \\ 1 & \text{if } e_{t-1}^2 = \text{continue and} \\ & \pi_t^2 = \pi_{t-1}^2 \\ 0 & \text{otherwise} \end{cases}$$

The level 1 behavior at time t depends upon the level 2 behavior at t and the level 1 behavior, state, and termination flag 1 at the previous timestep $t-1$. We define this as

$$P(\pi_t^1|\pi_t^2, \pi_{t-1}^1, s_{t-1}, e_{t-1}^1) =$$
$$\begin{cases} \sigma_{\pi_t^1}(\pi^2, s_{t-1}) & \text{if } e_{t-1}^1 = \text{terminate} \\ 1 & \text{if } e_{t-1}^1 = \text{continue and} \\ & \pi_t^1 = \pi_{t-1}^1 \\ 0 & \text{otherwise} \end{cases}$$

Termination flag 1 at time t depends upon the level 1 behavior and state at time t. We define this as

$$P(e_t^1|s_t, \pi_t^1) = \beta_{\pi_t^1}(s_t)$$

Termination flag 2 at time t depends upon the level 1 behavior, termination flag 1, and state at time t. We define this as

$$P(e_t^2|e_t^1, \pi_t^2, s_t) =$$
$$\begin{cases} \beta_{\pi_t^2}(s_t) & \text{if } e_t^1 = \text{terminate} \\ 0 & \text{otherwise} \end{cases}$$

The state at time t depends upon the level 1 behavior taken at time t and the state at time $t-1$.

$$P(s_t|s_{t-1}, \pi_t^1) = B(s_{t-1}, \pi_t^1, s_t)$$

C. Observation Model

The observation model signifies the probability of seeing an x,y position conditioned on a discrete hidden state. For this application the observations are modeled as a mixture of Gaussians. We explicitly model the mixture variable, m, as can be seen in Figure 1. The CPDs for this model can be written as follows.
For the observation nodes:

$$P(o_t|s_t = i, m_t = m) = N(o_t; \mu_{i,m}, \Sigma_{i,m})$$

For the mixture node:

$$P(m_t|s_t = i) = C(i, m)$$

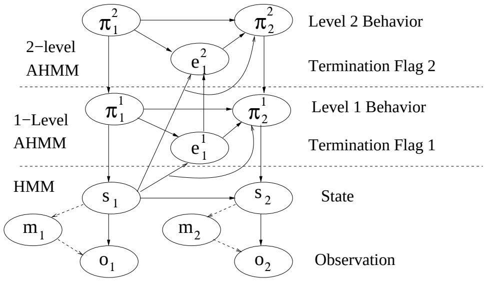


Fig. 1. DBN representation of a 2-level AHMM. The horizontal dashed lines indicate levels of hierarchy. For this specific application we use a mixture of Gaussians for the observation model. However this is not required by the model, thus we draw the links to and from the discrete mixture nodes (m_t) with a dashed line.

III. Inference and Learning In AHMM Models

The input to our models is a set of trajectories, $r = \{r_1...r_N\}$. Each trajectory consists of a sequence of observed positions $o_i = (x_i, y_i)$ which means that a trajectory $r_i = \{o_1, o_2, ..., o_T\}$. o_1 is the starting position and o_T is the final destination. Using these trajectories we are able to perform inference and learning in these models.

A. Inference

Inference can be thought of as querying the model. If X represents the node(s) of interest and E is the evidence, $P(X|E)$ is the query. In our model E is the trajectory r_i. X can be a node, for instance π_t^2 or s_t, or a conjunction of nodes, π_t^2 and s_t. Currently we use the junction tree (jtree) algorithm [7] for inference in the AHMM. Jtree is an algorithm for exact inference and calculates the marginals on all the nodes with a single forward-backward pass. However, Bui [5] presents a Rao-Blackwellised Particle Filter for approximate inference which is significantly more efficient.

There are different types of inference that can be performed, depending upon the type of questions asked of the model. If we return to our example there are several questions we might ask. As the person leaves the building we can ask which sub-behavior is currently being performed based upon the movements up until this time. Filtering is a type of inference that answers this type of question by recursively computing the belief state $P(X_t|o_1...o_t)$. For example, in our model we can calculate the belief state of an abstract behavior given the sequence of x, y positions observed thus far by calculating $P(\pi_t^k|o_1...o_t)$. If the person is walking down the stairs we can ask what behavior was performed at some previous time step given all the movement we have seen up until the current time. This type of question is answered by smoothing. Smoothing estimates past states while using all the evidence at our current time, $P(X_{t-l}|o_1...o_t)$. In our model we might ask for the abstract behavior performed at k timesteps in the past given all the current information we can calculate $P(\pi_{t-l}^k|o_1...o_t)$. If the person is currently in the hallway we can ask where they will be 3 timesteps into the future given all the information up until this time. Prediction estimates future states given the evidence available at the current time step, $P(X_{t+l}|o_1...o_t)$. Prediction allows us to predict what abstract behavior the agent will perform at some point in the future, $P(\pi^k_{t+1}|o_1...o_t)$. Or we could determine where the agent will be in future $P(s_{t+1}|o_1...o_t)$.

B. Learning

Since we do not handcode the parameters (CPTs or CPDs) of the models it is necessary to perform learning before inference can be done. We use EM to learn the parameters of our model.

1) Likelihood: Let θ represent the model parameters. Given the data D, the likelihood $P(D|\theta)$ can generally be computed from the joint distribution $P(X, D|\theta)$ by marginalizing (summing over) the hidden variables X in the model whose values are not defined by D. For the DBN representation of an AHMM, the likelihood can be computed by unrolling the DBN over the length of each trajectory. However, more efficient approximate inference methods are available, including variable elimination and particle filtering [5]. Let N denote the number of trajectories representing the evidence E and T_i represent the length of the ith trajectory. The joint distribution of the overall 1-level AHMM network given the parameters θ can be written as (the 2-level AHMM is specified by extending this case to include the extra level of hierarchy):

$$
\begin{aligned}
P(\pi^1, e^1, s, m, o|\theta) = {} & \\
& \prod_{i=1}^{N} P(\pi^1_{i,1}) P(e^1_{i,1}|\pi^1_{i,1}, s_{i,1}) \\
& P(s_{i,1}|\pi^1_{i,1}) P(m_{i,1}|s_{i,1}) \\
& P(o_{i,1}|m_{i,1}, s_{i,1}) \\
& \prod_{t=2}^{T_i} P(\pi^1_{i,t}|\pi^1_{i,t-1}, e^1_{i,t-1}, s_{i,t-1}) \\
& P(e^1_{i,t}|\pi^1_{i,t}, s_{i,t}) P(s_{i,t}|\pi^1_{i,t}, s_{i,t-1}) \\
& P(m_{i,t}|s_{i,t}) P(o_{i,t}|m_{i,t}, s_{i,t})
\end{aligned}
$$

2) Expectation Maximization: EM is a framework for maximum-likelihood parameter estimation with missing data [8]. If the data D are complete, i.e. if all the nodes in the model are observed, the maximum likelihood estimate of θ is $\hat{\theta} = \text{argmax}_\theta \log P(D|\theta)$, where D is the complete data. However, for training (1-level) AHMM models, the data D can be decomposed into an observed component S, and a hidden component U consisting of the nodes, π^1, e^1, s, and m that are not observed. EM finds the model parameters that maximize the *expected* value of the log-likelihood, where the data for the missing parameters are "filled in" by using their expected value given the observed data. EM consists of two steps, the Expectation step (E-step) and the Maximization step (M-step). The E-step computes the expected value of the log-likelihood, $Q(\theta|\theta^j) = E_{U|S,\theta^j}(logP(D|\theta))$, where the expectation is computed using θ^j, the model parameter estimate from the last iteration. For AHMMs, exact inference methods such as the junction tree algorithm can be used to determine the posterior distribution over the missing high-level nodes, given the observed trajectory. Alternatively, faster approximate methods based on particle filtering can also be used [5]. The M-step finds a new setting for the parameters such that $\theta^{j+1} = \text{argmax}_\theta \; Q(\theta|\theta^j)$.

3) Training on Labeled Data: We ran experiments where the models were trained with labels indicating which behavior was being performed. These labels were observed as the values of the highest level behaviors during training. Figure 2 shows both the 1-level and 2-level AHMMs used during training. Nodes that are shaded were observed during training. The models' performances were tested by performing inference to estimate the probability of the highest level behavior given test observations to see if the models distinguish the sequences by predicting the correct label.

4) Training on Unlabeled Data: We also trained the model on data where the highest level behavior was unobserved. Figure 3 shows the two types of models used when training with unlabeled data. When training with unlabeled data, we tested the ability of the models to cluster similar sequences together.

In order for the model to learn one label for each sequence at the highest level of behavior modeled it was necessary to fix the CPTs such that the highest level behavior never terminated. If the CPTs were not fixed the models learned behaviors that changed frequently over time. We fixed the CPTs for the 2-level AHMM as follows

$$
P(e^2_t|e^1_t, \pi^2_t, s_t) = \begin{cases} 1 & \text{if } e^2_t = \text{continue} \\ 0 & \text{otherwise} \end{cases}
$$

$$
\begin{aligned}
& P(\pi^2_t|\pi^2_{t-1}, s_{t-1}, e^2_{t-1}) = \\
& \begin{cases} 1 & \text{if } e^k_t = \text{continue} \\ & \text{and } \pi^2_t = \pi^2_{t-1} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

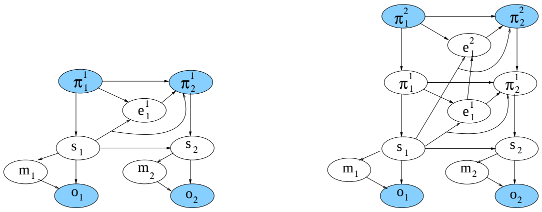
We fixed the CPTs for the 1-level AHMM as follows

$$
P(e^1_t|\pi^1_t, s_t) = \begin{cases} 1 & \text{if } e^1_t = \text{continue} \\ 0 & \text{otherwise} \end{cases}
$$

$$
\begin{aligned}
& P(\pi^1_t|\pi^2_t, \pi^1_{t-1}, s_{t-1}, e^1_{t-1}) = \\
& \begin{cases} 1 & \text{if } e^1_t = \text{continue} \\ & \text{and } \pi^1_t = \pi^1_{t-1} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

IV. Experiments

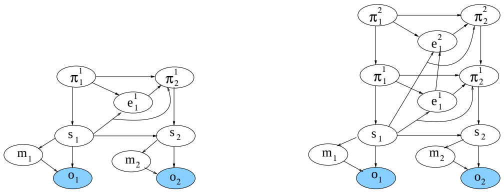
We performed several different types of experiments to evaluate the AHMM as a framework for hierarchical activity modeling. We limited our models to the 1-level AHMM and the 2-level AHMM. We tested the models on three different datasets. Experiments were run using the Bayes Net Tool Box for Matlab (BNT) [9].



(a) 1-level AHMM with observed level 1 behavior nodes.

(b) 2-level AHMM with observed level 2 behavior nodes.

Fig. 2. Models trained with labeled data. The shaded nodes are observed.



(a) 1-level AHMM with unobserved level 1 behavior nodes

(b) 2-level AHMM with unobserved level 2 behavior nodes

Fig. 3. Models trained with unlabeled data. The shaded nodes are observed.

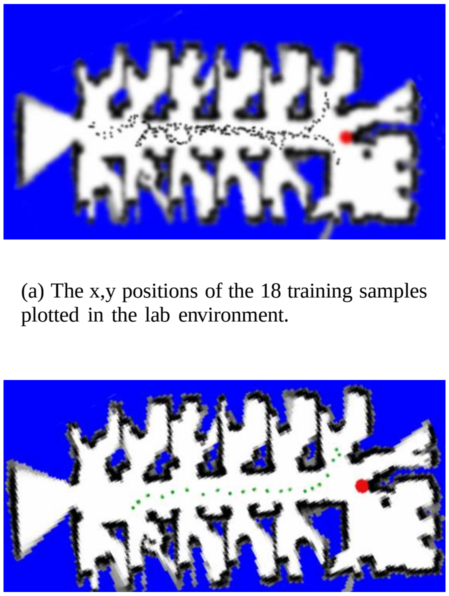
A. Lab Data

Our first dataset was collected using Hema, a B21r robot equipped with a laser range finder. Hema was positioned so she could view most of the activity occurring within the hallway of a lab environment. Laser readings were collected using CARMEN [10]. Figure 4 shows the lab environment: Hema was positioned at the end of the long corridor. Laser scans of people walking along different trajectories through the lab were collected. An example trajectory can be seen in Figure 4(b).

The laser scans were processed to extract the (x,y) position of each person walking. This was achieved by first computing the difference between scans at time t and t' and finding the degree where the greatest change between the two scans occurs. The position was calculated using the robot's localized position on the map. Every third time slice was used to smooth the trajectory.

There were 6 different classes of sequences collected in the lab. The models were trained using 3 instances of each sequence. The test set consisted of three previously unseen instances. The sequences differed in length, consisting of anywhere from 6 to 24 (x,y) pairs. Figure 4(a) shows the 18 training sequences plotted in the lab environment.

1) Model Definition: We found that setting the observation model to one mixture component gave the best performance. All models were initialized using six values for highest level behavior multinomial nodes and 50 values for the discrete multinomial state nodes. Often, far fewer states were needed to represent the data as seen in Figure 6 where only 15 states were used. For the experiments with the 2-level AHMM, level 1 behavior nodes were initialized to have 15 values. The highest level behaviors were initialized to have a uniform probability. The means of the Gaussians were initialized to be random points in the data and the covariance matrices were initialized to identity. All other parameters were initialized randomly.



(a) The x,y positions of the 18 training samples plotted in the lab environment.

(b) An example of a single trajectory. In this example the person walks from one cube near the door toward the robot until it goes into another cube.

Fig. 4. Data collection within the lab environment

B. Entryway Data

One of the characteristics of the data in the lab environment was that the trajectories overlapped for most of their duration. Data were collected in the entryway of the 2nd floor of the computer science building where typical paths of motion were more distinct. The data are shown in Figure 5. Data collection and processing was the same as the lab data in the previous section.

The dataset consisted of eight different types of sequences. The training set consisted of eight instances of each sequence; the test set contained 2 instances. The sequences differed in length, consisting of between 15 to 30 (x,y) pairs.

1) Model Definition: Once again we found that setting the observation model to one mixture component gave the best results. All models used eight values for the highest level behavior nodes and 60 values for state nodes, although approximately 30 were typically used. For the

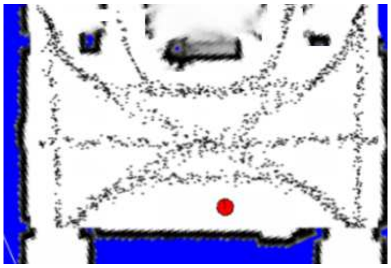


Fig. 5. Data collected in the entry to the 2nd floor of the UMass CS department. The data consists of 8 different classes of trajectories. The location of the robot is indicated by the filled circle.

experiments with the 2-level AHMM, level 1 behaviors had 25 possible values. The highest level behaviors were initialized uniformly. As above, the means of the Gaussians were initialized to be random points in the data and the covariance matrices were initialized to identity. All other parameters were initialized randomly.

C. Home Data

We also ran experiments on data collected by Bennewitz et al [4]. The data consist of sequences of a person moving around a house. The data were collected using three Pioneer I robots equipped with laser range finders.

This data set has 11 different sequences with three instances for each sequence type. Due to the small amount of data, we used cross-validation to evaluate the performance of the models.

1) Model Definition: We found that setting the observation model to a single mixture component gave the best results. All models had 11 values for the highest level behavior nodes and state nodes had 50 values of which 17 were approximately used. For the experiments with 2-level AHMMs, level 1 behaviors could take on 17 values. The highest level behaviors were initialized with uniform probability. Once again, the means of the Gaussians were initialized to be random points in the data and the covariance matrices were initialized to identity. All other parameters were initialized randomly.

V. Results

The performance of each model was determined using the percent of test sequences that were correctly classified when the model was run on unseen test data. These results are shown in Table I. This table shows that in all cases the 2-level AHMM performs as well, if not better, than the 1-level AHMM. The models trained with labeled data learn to distinguish the trajectories better than the models found using unlabeled data. The results for the unlabeled data show how the levels of hierarchy make a difference. The 2-level AHMM performed better in the cases where there was more overlap between the trajectories, such as lab there where the overlap made it more difficult to distinguish between the trajectories. The entryway data has some overlap but overall the trajectories differ over much of their duration. It may be possible to gain better prediction results in the models trained with unlabeled data by biasing the values of the parameters and hidden nodes.

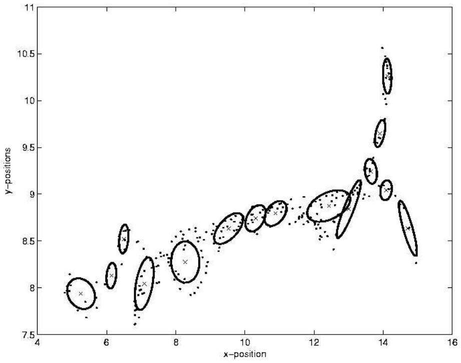


Fig. 6. The learned observation distributions $P(o_t|s_t, m_t)$. Each ellipse represents the covariance matrix of the Gaussian for each given state s_t. The Gaussians are overlaid on the (x,y) points that make up the data. This set of states was learned using a 2-level AHMM.

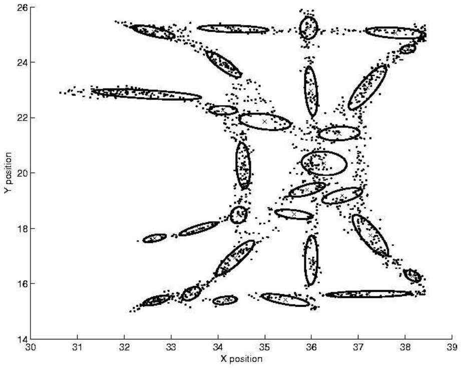


Fig. 7. The learned observation distributions $P(o_t|s_t, m_t)$. Each ellipse represents the covariance matrix of the Gaussian associated with a given state s_t. The Gaussians are overlaid on the (x,y) points that make up the data. This set of states was learned using a 2-level AHMM.

After training the observation model, $P(o_t|s_t, m_t)$ was plotted on top of the (x, y) positions of the data. The observations model clustered areas where motion took place within the environment. Figure 6 shows the observation model for the lab data and Figure 7 shows the same plot for the entryway data.

We also performed inference to test the models ability to predict higher level behavior. Figure 9 shows the results of filtering in a 2-level AHMM trained on labeled data for a trajectory in the entryway. The plot shows the probability of the level 2 behavior at each time given the current sequence of observations, $P(\pi_t^2|o_1...o_t)$. Figure 8 shows the same results except in the model trained with unlabeled data. Both graphs show that the likelihood of the correct class goes up as the trajectory progresses, distinguishing itself from other trajectories that share overlapping portions.

TABLE I

PERCENTAGE OF TEST SEQUENCES CORRECTLY CLASSIFIED AFTER TRAINING. RANDOM GIVES RESULTS FOR GUESSING WITH EQUAL PROBABILITY FOR EACH BEHAVIOR.

Model	Lab	Entryway	Home
Random	16.67%	12.5%	9.09%
1-level AHMM with unobserved Level 1 behavior	66.67%	75%	57.57%
1-level AHMM with observed Level 1 behavior	100%	100%	100%
2-level AHMM with unobserved Level 2 behavior	83.33%	75%	60.60%
2-level AHMM with observed Level 2 behavior	100%	100%	100%

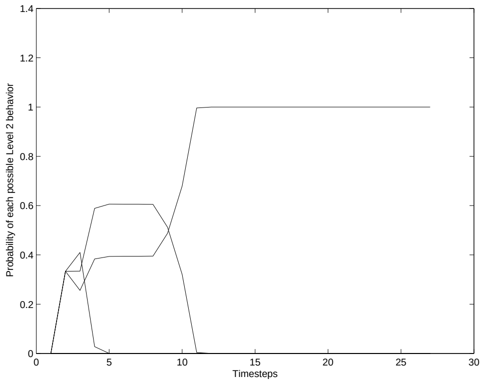


Fig. 8. The probability of the level 2 behavior at each time given the current sequence of observations, $P(\pi_t^2|o_1...o_t)$ plotted for one trajectory recorded in the entryway. The model was a 2-level AHMM trained with labeled data.

VI. CONCLUSION AND FUTURE WORK

In this paper we presented a hierarchical approach to modeling motion behavior in an indoor environment. We compared 1-level and 2-level AHMMs, where the parameters of the models were learned using EM. We show that hierarchical models outperform flat models in cases when classification is especially difficult. We also compared both supervised and unsupervised learning in these model.

There are several areas for future work. Currently we have only investigated the case where one person is moving through the environment, and extending it to the multi-agent case can be addressed using the multi-agent AHMM model proposed in [11]. Other open questions involve structure learning and model selection for AHMMs. We assumed that the number of abstract behaviors, and states were known, as well as the number of levels in the hierarchy. Efficient approaches to model selection for AHMMs remains an open problem to be investigated. Parameter estimation using exact inference is expensive. We are currently exploring approximate techinques. Further experiments comparing 1-level and 2-level models should be performed to better determine conditions where hierarchy helps improve classification accuracy.

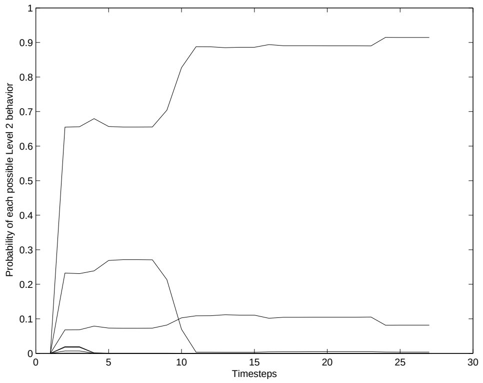


Fig. 9. The probability of the level 2 behavior at each time given the current sequence of observations, $P(\pi_t^2|o_1...o_t)$ plotted for one trajectory recorded in the entryway. The model was a 2-level AHMM trained with unlabeled data.

ACKNOWLEDGMENT

This research is supported in part by Michigan State University under subcontract #61-3546A (via DARPA Contract Number DABT63-99-1-0014). Victoria Manfredi is supported by an NSF Graduate Fellowship.

The authors would also like to thank the members of Autonomous Learning Lab who generously participated in these experiments.

REFERENCES

[1] M. Montemerlo, S. Thrun, and W. Whittaker, “Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-tracking,” in *IEEE Int. Conf. on Robotics and Automation*, May 2002.

[2] A. Fod, A. Howard, and M. Matarić, “Laser-based People Tracking,” in *IEEE Int. Conf. on Robotics and Automation*, May 2002.

[3] H. Yan and M. Matarić, “General Spatial Features for Analysis of Multi-robot and Human Activities from Raw Position Data,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Switzerland, October 2002, pp. 2770–2775.

[4] M. Bennewitz, W. Burgard, and S. Thrun, “Using EM to Learn Motion Behaviors of Persons with Mobile Robots,” in *Int. Conf on Intelligent Robots and Systems*, 2002.

[5] H. Bui, S. Venkatesh, and G. West, “Policy Recognition in the Abstract Hidden Markov Model,” *Journal of Artificial Intelligence Research*, vol. 17, pp. 451–499, Dec 2002.

[6] K. Murphy, “Dynamic Bayesian Networks: Representation, Inference and Learning,” Ph.D. dissertation, University of California, Berkeley, 2002.

[7] C. Huang and A. Darwiche, “Inference in belief networks: A procedural gude.” *International Journal of Approximate Reasoning*, vol. 15(3), pp. 225–263, 1996.

[8] A. Dempster, N. Laird, and D. Rubin, “Maximum-likelihood from Incomplete Data via the EM Algorithm,” *Journal of Royal Statistical Society Series B*, 1977.

[9] K. Murphy, “The Bayes Net Toolbox for Matlab,” *Computing Science and Statistics*, vol. 33, 2001.

[10] M. Montemerlo, N. Roy, and S. Thrun, “Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) toolkit,” in *International Conference on Intelligent Robots and Systems*, 2003.

[11] S. Saria and S. Mahadevan, “Online probabilistic plan recognition in multiagent systems,” in *Fourteenth International Conference on Automated Planning and Scheduling*, June 2004.